Oliver Krauss (presenting) and William B. Langdon

Automatically Evolving Lookup Tables for Function Approximation

EuroGP (evostar) 15. April 2020

Abstract

- Big picture: Genetic Improvement to *automatically create or update constants* in software
- This publication: *creation of lookup tables* for methods that can be *approximated*
- Contribution: *high precision* and *fast run-time* performance.
- Results: *double precision* accuracy outperforming reference implementations
- Limitations: *function range* and functions with *inflection points*

Introduction

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{1}$$

- Newton-Raphson (1) approximation
- Improve run-time and accuracy with lookup table for x_0
- Restrict amount of iterations to *3*

Contributions

- Double precision accuracy (1 sign bit, 11 bit exponent, 52 bit fractional [1])
- Automated generation via user provided function
- Fast run-time performance compared to not hardware-accelerated functions

Background - Covariance Matrix Adaptation - Evolution Strategy (CMA-ES)

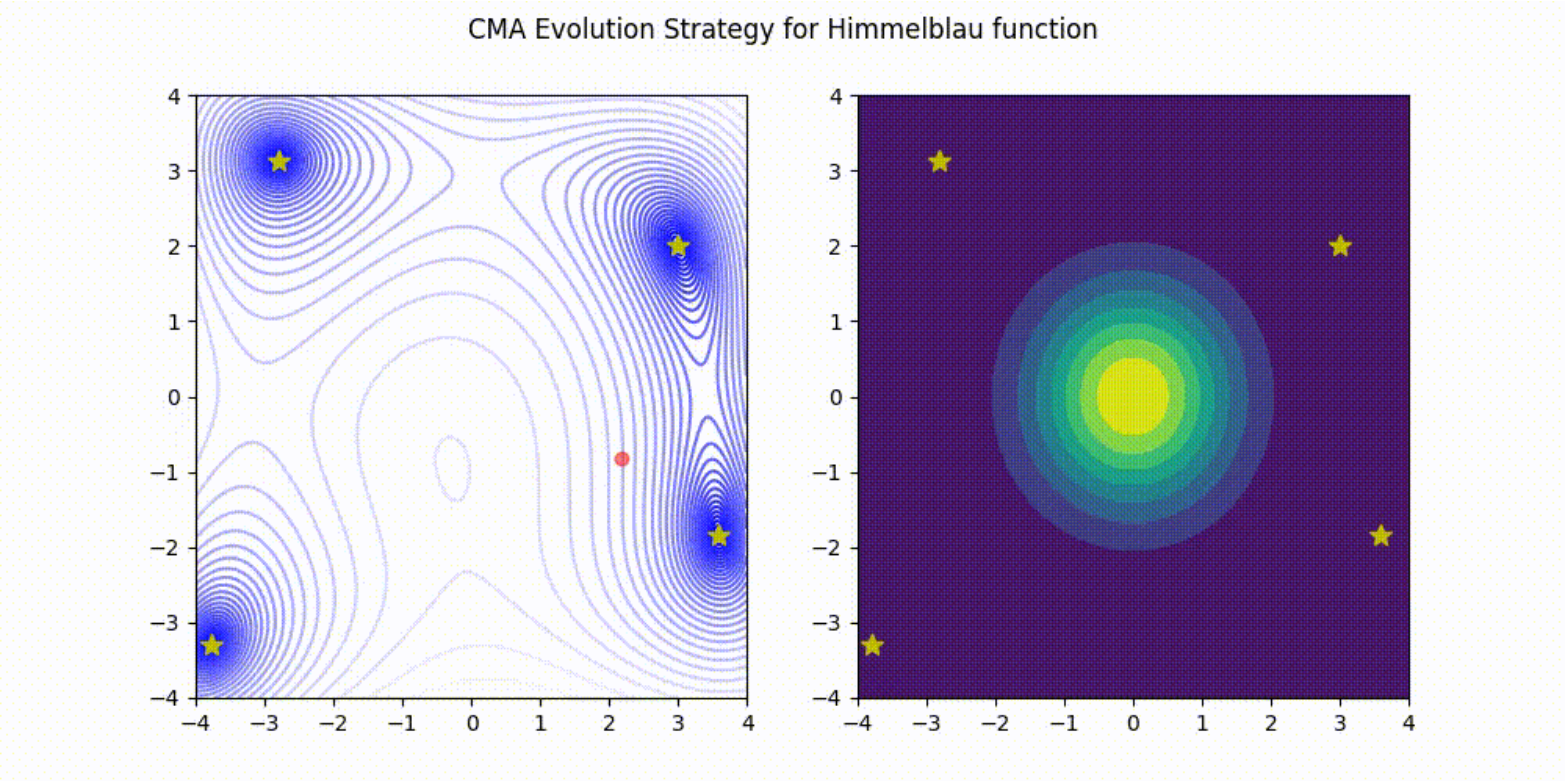


Figure: CMA-ES evolution of population (left) around centroid (right) [2]

Reporting Test Results

For all following evaluations

- Testsets in a range between 0.5 and 2
  - Exception next slide - range between 0.5 and 10,000
- 512 values even distribution (0.5, 0.5029, 0.5058, ...)
- 512 values random
- Random Seed for CMA-ES between 1 and 1,000,000 (random selection)
- $err = abs(fn(approximation) - input)$

Background - Evolving better Software Parameters

- Previous work by Langdon and Petke [3]
- glibc - square root adapted to cube root
- lookup table generated with CMA-ES
- Fitness function adapted with log - shown to not matter
- Outperforms Java and C++

Implementation	Distribution	Total Error $\times$ 10^{-10}
C - with log	even	**3.1451**
	random	***3.3231***
C - without log	even	**3.1451**
	random	***3.3231***
Java	even	3.3322
	random	3.6071
C++	even	6.2851
	random	7.2275

Methods - Generalization

```
// Function for Newton-Raphson
double fn(const double approx) {
    return approx * approx * approx;
}

// Derivative of fn
double derivativeFn(const double approx) {
    return 3 * approx * approx ;
}

// Function that allows user to modify input and result
double userFunction(const double x) {
    if (x < 0) return approximate( val: 0.0 - x);
    if (x > 0) return approximate(x);
    return x;
}
```

Figure: Generalization via user provided function and derivative

Methods - Fitness Function I

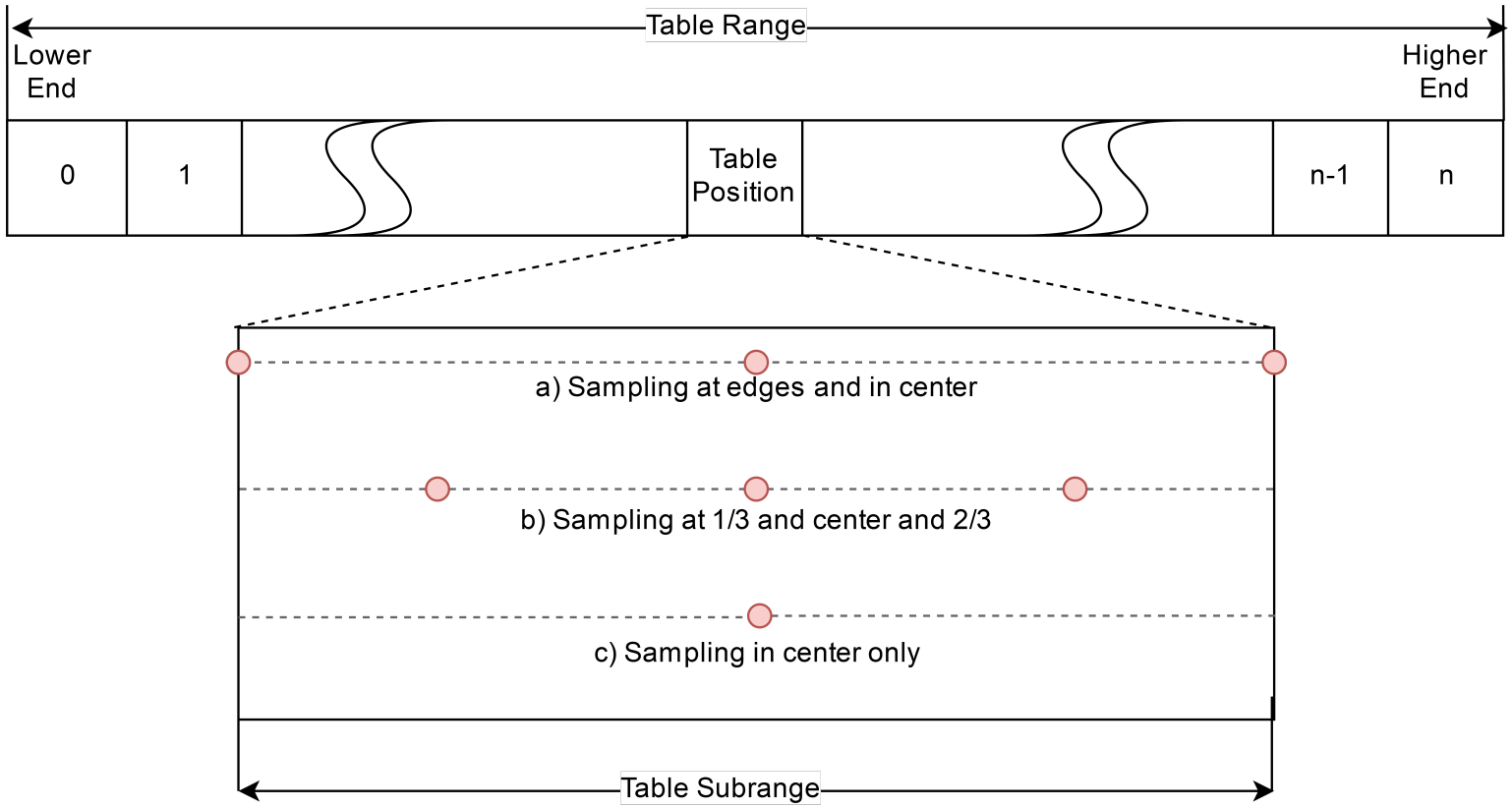
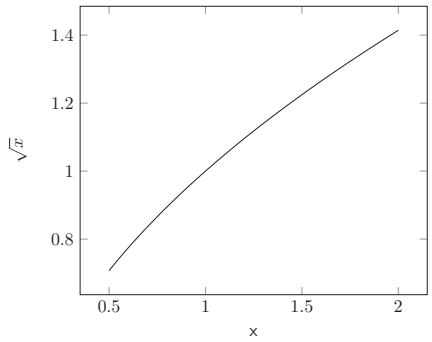


Figure: Sampling points for fitness function

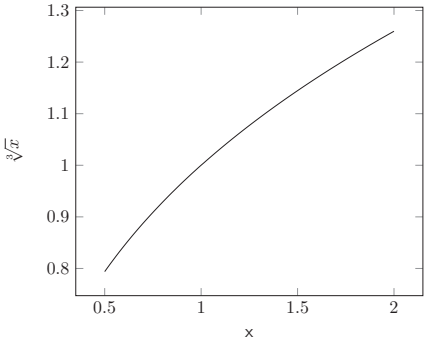
Methods - Fitness Function II

- Approximation -
  $fitness = abs(fn(newtonRaphson(input)) - input)$
- Remaining Error -
  $fitness = remainingErrorNewtonRaphson(input)$
- Direct - $fitness = abs(fn(tableEntry) - input)$

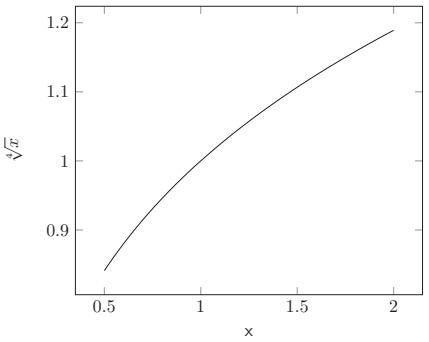
Results - Root and Polynomial Functions



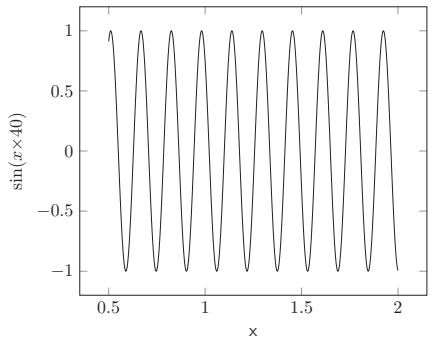
(a) Square root $\sqrt{x}$



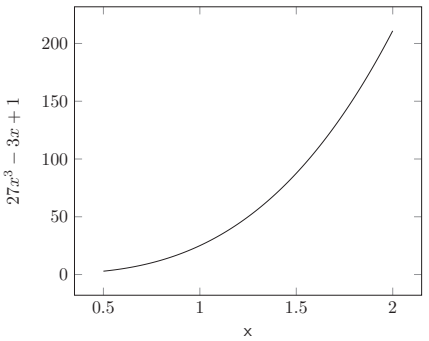
(b) Cube root $\sqrt[3]{x}$



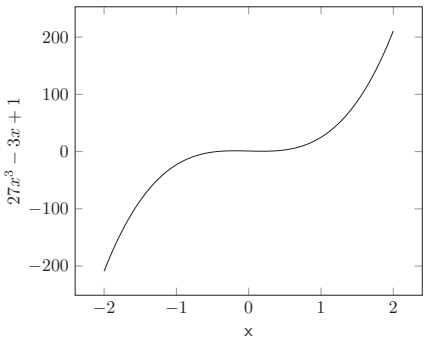
(c) Super root $\sqrt[4]{x}$



(d) Multiple Inflections $\sin(x\times40)$



(e) Polynomial with inflection $27x^3 - 3x + 1$



(f) Polynomial with inflection

Results - Accuracy

Fitness Function

- In even distribution sampling at edges works best
- In random distribution sampling only center works best
- Applying Log (and other operations) to fitness function improves outcome
- On inflections, some fitness functions prevented convergence of CMA-ES

Comparison to Java, C, C++

- Slightly less accurate in Square Root
- Outperforms C++ and Java in Cube Root and Super Root

Results - Run-Time Performance

Table: Run-time performance of root functions (average of 1,000,000 calls)

	Mean (in nanoseconds per call)		
Language	sqrt	cbrt	surt
Hardware Accelerated C	0.88	0.88	0.88*
Hardware Accelerated C++	4.10	21.35	8.10*
Langdon and Petkes cbrt		24.96	
Our approach	25.33	27.46	29.58
Java	1.02	69.51	1.03*

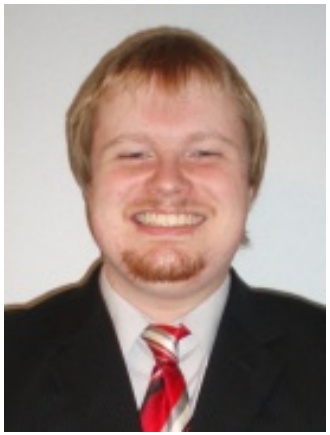
*surt implemented as sqrt(sqrt(x))

Conclusion and Outlook

- Use Case for functions that have no hardware acceleration
- Achieving *double precision* and fast *run-time* with *smooth functions*
  - Limited to *specific range*
  - Does not work with inflections
  - Robustness of function has to be considered
- Big Picture: Fitness function design imperative for updating and generation of constants

Questions?

Code and data available at
https://github.com/oliver-krauss/EuroGP2020-LookupTables.



Oliver Krauss
Johannes Kepler University Linz
University of Applied Sciences Upper Austria
Oliver.Krauss@fh-hagenberg.at
Homepage: *http://aist.fh-hagenberg.at*

William B. Langdon
University College London
w.langdon@cs.ucl.ac.uk
Homepage: *http://www0.cs.ucl.ac.uk/staff/W.Langdon/*

Bibliography I

[1] IEEE, "Standard for Floating-Point Arithmetic", *Std 754-2008*, Aug. 2008.

[2] CyberAgent, *cmaes*, [Online; accessed 14. Apr. 2020], Apr. 2020. [Online]. Available: https://github.com/CyberAgent/cmaes.

[3] W. B. Langdon and J. Petke, "Evolving Better Software Parameters", in *Search-Based Software Engineering*, Springer, 2018, pp. 363–369.